UNIVERSITAT DE BARCELONA

Treball final de grau

GRAU D'ENGINYERIA INFORMÀTICA

Facultat de Matemàtiques
Universitat de Barcelona

USER BEHAVIOUR ANALYSIS ON REDDIT

Author: Huang Chen

Supervisor: Dr. Lluís Garrido Ostrermann
Affiliation: Departament de Matemàtica Aplicada i Anàlisi. UB
Barcelona, 30 de juny de 2016

Contents

Acknowledgement

First and foremost, I would like to express my sincere gratitude to my advisor Dr. Lluís Garrido Ostrermann, for his immense knowledge, the valuable support he has given me during the entire project, as well as he patiently offered me skype meetings when I was studying abroad. I am really thankful to him.

Secondly, I would also like to thank my parents for the continuous supports they have been giving me.

Finally, I would also like to extend my thanks to my friends and double degree classmates for the motivation they have given me during all the degree.

Chapter 1

Introduction

1.1 Motivation

Nowadays, Big Data and Data Science seem to be the most useful technologies for data processing used in a variety of areas, such as business, sports, social media, investigations, etc.

But, what is Big Data? And, what about Data Science? Also, what is the difference between them?

Big data is the challenge for data sets that are huge or complex which traditional data processing applications are inadequate to manage them in a reasonable time[1]. Hence, we can conclude Big data as a concept that refers to the storage of large amounts of data and the used procedures to find repetitive patterns within the data.

Data Science is an interdisciplinary field about processes and systems to extract knowledge or insights from data in various forms, either structured or unstructured, which is continuation of some of the data analysis fields such as statistic, data mining, and predictive analytics, similar to Knowledge Discovery in Database(KDD).

We compare them. Data Science looks to create models that capture the underlying patterns of complex systems, and codify those models into working application. Big Data looks to collect and manage large amounts of varied data to serve large-scale web applications and vast sensor networks. But, these terms are often used interchangeably despite having fundamentally different roles to play in bringing the potential of data to the doorstep of an organization[7].

As social media is now the fastest way to spread out information and it has grown up quickly in these recent years. So, it is interesting to know how is the database behind social media being processed. That is the reason I have been led to this topic and chose it for my Degree Final Project (TFG).

Now, how are Big Data and Data Science applied in social media?

Social media is the collective of online communications channels dedicated to com-

munity based input, interaction, content-sharing and collaboration. In other words, it is computer-mediated tool that allow people exchange information in virtual communities. Hence, it must have an enormous and complex database, because that information can be created and stored by multi formats files. For example, pictures, videos, texts, etc. Then, in pictures category, it has various extensions like PNG, JPEG, GIF, etc. Therefore, we need apply Big Data and Data Science concepts to manage and process them intelligently.

However, in this project we are going to talk about just the small particular piece of social media, which is Reddit.

1.2 Context

Reddit is an entertainment, social news website founded by Steve Huffman and Alexis Ohanian in June 23rd, 2005. Registered users on Reddit are called redditors, they can submit content as text posts or direct links, then other redditors can post comments about it or reply other comments, and vote submission with "like" or "unlike" options. That is, the submissions with the highest amount of "like" appear on the main page or the top of category. Likewise, redditors can vote negatively to against the topic, that can probably make it disappear from the hottest posts. Contents entries are categorised by dividing into different areas of interest called "subreddits"[4].

Reddit has around 50 default subreddit topics like news, gaming, movies, music, and many others which are visible in front of page. More than that, redditors can also create their own subreddit topic.

Moreover, Reddit is one of the most popular social media website in the world. There are 36 million redditors and 231 million monthly uniques in Reddit counted until February 19, 2016. Therefore, we can conclude that Reddit has a huge database which needs some special technology such as Big data to process them easily.

1.3 Aim of Degree Final Project(TFG)

As we mentioned in the above section, Reddit has a large amount of visitors, even though there is a piece of them are non-login users. But it still has huge number of redditors. However, no all of them behaviour well and keep Reddit as a clean virtual social community. In other words, some of Reddit visitors post text which contains lots insults or they rudely comment posts. Therefore, we aim to analyse the redditors posts and find out the worst behaviour user on Reddit during whole month in May of 2015.

However, we do not have any stored information about non-login users except their comments. So, we are going to point out just the worst behaviour redditors. That is, analyse Reddit database of May, 2015, then process them by using an appropriate methodology to calculate the "bad" rate of each user. Such "bad" rate is an integer number between 0 to 100, which higher "bad" rate gets, worse behaviour that redditor has. Then, we sort

the result by using "bad" rate list to get the highest "bad" rate users. Those redditors are supposed to be the worst behaviour users in Reddit.

1.4 Report overview

In order to achieve the above goal, we organize this project as follows:

Chapter 2 explains the used methodology and the chosen programing languages.

Chapter 3 discusses the implementation of program, which includes: how we extract information from the database; what methodology we used to analyse redditors' behaviour; how we added parallelism concept to make program accelerate execution speed for such huge database.

Chapter 4 shows experimental results with some histogram, DataFrame tabular. Also, it details the improvement of execution speed by using Parallelism.

Chapter 5 ends the project with conclusion and propose other possible future works.

Chapter 2

Technologies

In fact, there are various programing languages to perform this project. However, the four main languages for Analytics, Data Science, Data Mining are the following: R, SAS, SQL, Python. The following Venn diagram shows the popularity between these four programming languages voted by users of KDnuggets in 2014, which KDnuggets is a leading site on Business Analytics, Data Mining, and Data Science[8].

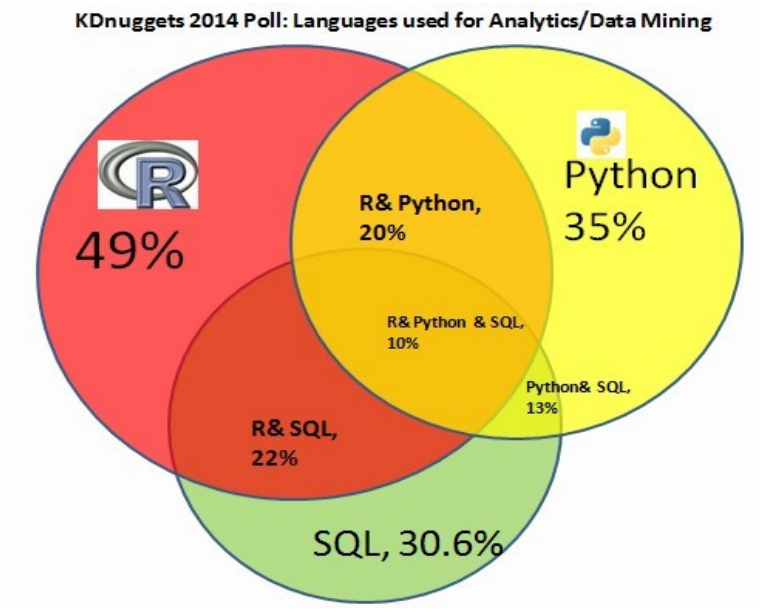


Figure 2.1: Proportion of languages used for Analytics/Data Mining between R, Python, SQL.

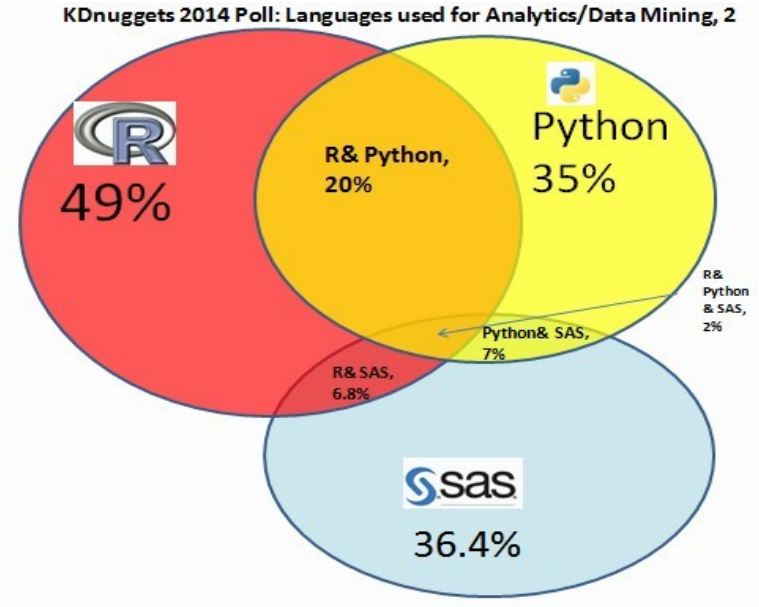


Figure 2.2: Proportion of languages used for Analytics/Data Mining between between R, Python, SAS.

Among these four languages, R and Python got most of votes. Hence we are going to discuss in which language among them suits us better to perform this project.

2.1 R vs Python

2.1.1 History about R and Python

R is a programming language and software environment for statistical computing, it is also the open-source language which was created by Ross Ihaka and Robert Gentleman at the University of Auckland in 1995 as an implementation of the S programming language. At first, R was primarily used in academics and researches, but the enterprise is discovering it as well. That makes R become one of the fastest growing statistical language in the corporate world. The first purpose of creating R language was to develop a language for a better and more user-friendly way to analyse data, and build statistics graphical models.

Python is a widely used high-level, general-purpose, interpreted, dynamic programming language which was created by Guido Van Rossem in 1991 as a successor to the ABC language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than possible in languages such as C++ or Java. The main users of Python are those programmers that want to delve into data analysis or apply statistical techniques for statistical purposes.

2.1.2 Why Python?

On the web, we can find easily many numbers comparing the adoption and popularity of R and Python. Whilst these give a good indication on how these two languages are evolving in the overall ecosystem of Computer Science, it is hard to compare them side-by-side. The main reason for this is that R seems to be just in a Data Science environment; However, Python as a general purpose language, it is widely used in many field, such as web development. This often biases the ranking results in favour of Python.

Moreover, R is mainly used when the data analysis task requires standalone computing or analysis on individual servers. It is handy for almost any type of data analysis because of the huge number of package and readily usable tests that often provide us with the necessary tools to get up and running quickly. R can even be part of a Big Data solution.

Contrast to R, we can use Python when data analysis tasks need to be integrated with web applications or if statistics code needs to be incorporated into a production database. Being a fully-fledged programming language, it is a great tool to implement algorithms for production use. While the infancy of Python packages for data analysis was an issue in the past, this has improved significantly over the year.

The comparison between R and Python can obtained by list the pros and cons of both[9].

The advantages of R are the following:

- R can effectively visualize data. It contains a powerful visualization packages such as ggplot2, ggvis, googleVis and rCharts.
- R has a rich ecosystem of cutting-edge packages and active community. Packages are available at CRAN, BioConductor and Github. We can search through all R packages at Rdocumentation.
- R is developed by statisticians. They can communicate ideas and concepts through R code and packages, hence you do not necessarily need a computer science background to get started.

Despite the above such useful advantages for analytics of data, but we have also take into account some disadvantages of it.

- R is slow. That is because R was developed to make the life of statisticians easier, hence R can be experienced as slow due to poorly written code.
- Learning R is difficult, especially if you come from a GUI for your statistical analysis. Even finding package can be time consuming if you are not familiar with it.

Now, Python has the following advantages in data analysis:

- Python is a general purpose language that is easy and intuitive. Hence its learning curve is relatively low, and it increases the speed at which we can write a program.

In short, we need less time to code. Furthermore, the Python testing framework is a built-in, low-barrier-to-entry testing framework that encourages good test coverage. This guarantees the programming code is reusable and dependable.

- Python is a multi-purpose language. It brings people with different backgrounds together. As a common, easy to understand language that is known by programmers and that can easily be learnt by statisticians, we can build a single tool that integrates with every part of our workflow.

- One of the strengths of Python is the IPython Notebook. It makes us easier to work with Python and data. We can easily share notebooks with other people, without having them to install anything. This drastically reduces the overhead of organizing code, output and notes files. It will allow us to spend more time doing real work.

Comparing the disadvantages of Python to R:

- Although Python has some nice visualization libraries, such as Seaborn, Bokeh and Pygal, there are maybe too many options to choose from. However, compared to R, visualizations are usually more convoluted, and the results are not always so pleasing to the eye.

- Python is a challenger to R. It does not offer an alternative to the hundreds of essential R packages. Although it is catching up, it is still unclear if this will make people give up R.

Regarding to the above comparison, it seems the cons of Python are more bearable than R. That is the reason we chose Python as the programming language for this project.

Moreover, Python has been evolved from Python 1.x version to Python 3.5.1 up to the present day. But the mostly Python 1.x versions have been deprecated. Hence, the description of some features about Python 2.x and Python 3.x are below.

Python 2.0 was released on 16 October 2000 and had many major new features, including a cycle-detecting garbage collector and support for Unicode. With this release the development process was changed and became more transparent and community-backed.

Python 3.0 was released on 3 December 2008 after a long period of testing. Many of its major features have been backported to the backwards-compatible Python 2.6.x and 2.7.x that is now the earliest still supported version.

In fact, Python 3.0 version was firstly used, but it has some incompatible code in the Parallelism part which we are going to talk on the next section. Therefore, after trying Python 3.0, it is replaced by Python 2.7.

2.2 Ipython Notebook and Parallelism

IPython is a command shell for interactive computing in multiple programming languages, originally developed for the Python programming language, that offers introspection, rich media, shell syntax, tab completion, and history. IPython provides features as follows: Interactive shells; Support for interactive data visualization and use of GUI toolkits; Flexible, embeddable interpreters to load into one's own projects; A browser-based notebook with support for various media; Tools for parallel computing[2]. Regarding to all these features, we just focus on IPython Notebook and IPython Parallelism.

2.2.1 IPython Notebook

As we mentioned Ipython Notebook at the previous section. It is a web-based interactive computational environment for creating IPython notebooks. An IPython notebook is a JSON document containing an ordered list of input/output cells which can contain code, text, mathematics, plots and rich media.

In addition, IPython notebooks can be converted to a number of open standard output formats through its web interface and shell. These output formats can be HTML, HTML presentation slides, LaTex, Markdown, Python, etc.

Thus, we can conclude an IPython notebook as a tool which lets us write and execute Python code in our web browser. Also, IPython notebooks make it very easy to tinker with code and execute it in bits and pieces; for this reason, IPython notebooks are widely used in scientific computing.

2.2.2 IPython Parallelism

Parallelism has been employed for many years, mainly in high-performance computing. Parallel computing is a type of computation in which many calculations are carried out simultaneously, in order to increase the execution speed and reduce execution time. Operating on the principle that large problems can often be divided into smaller ones, which are then solved at the same time.

IPython has a powerful architecture for parallel and distributed computing. This architecture abstracts out parallelism in a very general way, which enables IPython to support many different styles of parallelism including: Single program, multiple data parallelism; Multiple program, multiple data parallelism; Message passing using MPI; Task farming; Data parallel; Combinations of these approaches; Custom user defined approaches. Most importantly, IPython enables all types of parallel applications to be developed, executed, debugged and monitored interactively.

The IPython architecture consists of four components which live in the IPython.parallel package and are installed with IPython. These components are structured as the shown diagram[5].

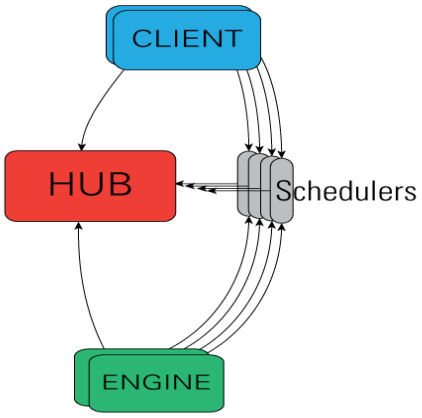


Figure 2.3: IPython's architecture for parallel computing

Each of the blocks explained below:

- The IPython engine is a Python instance, usually a IPython interpreter, that takes Python commands over a network connection. When multiple engines are started, parallel computing becomes possible.
- A HUB and a collection of schedulers form a IPython controller. The controller is a collection of processes to which IPython engines and clients can connect, it provides an interface for working with a set of engines.
- HUB is the centre of an IPython cluster. It is also the process that keeps track of engine connection, schedulers, clients, as well as all task requests and results. It has to facilitate queries of the cluster state, and minimize the necessary information required to establish the many connections involved in connecting new clients and engines.
- The scheduler is an application that distributes the commands to the engines. While the IPython engines themselves block when user code is run, the schedulers hide that from the user to provide a fully asynchronous interface to a set of engines.
- The client is a IPython object created at an IPython interpreter. This object will allow us to send commands to the IPython engines.

2.2.3 Multicore parallel computing

In this project, we use the multicore parallel computing of IPython parallelism. A simple concept to understand it is, to split the computation work in multiple tasks so that

each one is executed in different cores.

Assume that a task takes T seconds to run on a single core (using standard serialized programming). Now assume that we have a computer with N cores and that we have divided our serialized application into N subtasks. By using the parallel capabilities of our computer we may be able to reduce the total computation time to T/N. This is the ideal case and usually we will not be able to reduce the computation time by a factor of N. This is due to the fact that cores, on one hand, need to synchronize between them at the hardware level in order to access common resources such as the RAM memory and, on the other hand, the operating system needs some time to switch between all the tasks that run on the computer. However, using the multicore capabilities of the computer unit will result in an improvement of the computation time if the tasks are properly defined.[6]

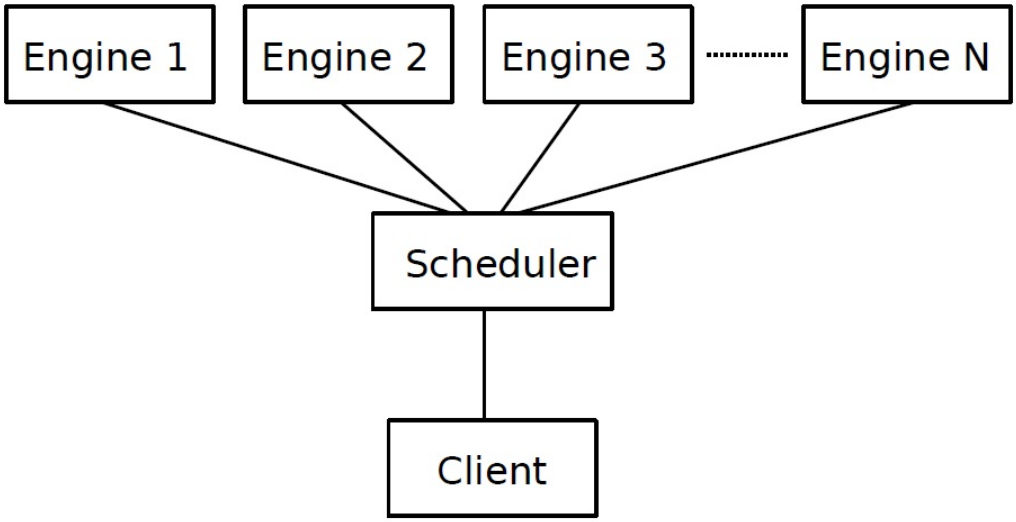


Figure 2.4: IPython's architecture for multicore parallel computing.

A simplified version of the IPython's architecture for multicore parallel computing is shown above.

IPython uses the term cluster to refer to the scheduler and the set of engines that make parallelization possible. In addition, we should take into account that each engine is an independent instance of a IPython interpreter. Hence, it is an independent process. Moreover, we may be able to control at which engine each task is executed but we will not be able to control on which core each engine is executed.

In order to perform Multicore programming, IPython controller provides two primary models for interacting with engines:

1. A Direct View interface, where engines are addressed explicitly.
2. A Load Balanced interface, where the scheduler is trusted with assigning work to appropriate engines.

During this project, we use Direct View on account of its simplicity. It allows the user to directly control which tasks are sent to which engines.

2.2.4 Connecting to the cluster

To use IPython's parallel capabilities the first thing to do is to start the cluster. There are two ways for doing it. We just detail the simplest way which is started from the notebook interface. Within the IPython notebook, we can use the "Clusters" tab of the dashboard, and press "Start". This will automatically run the necessary commands to start the IPython cluster creating N engines, where N equals the N number of cores. In this case the notebook will be used as interface to the cluster, that is, we will be able to send diverse tasks to the engines using the web interface.

We have seen how to initialize the cluster with N engines. Now, we use the following command to connect it.

```
In [1]: from IPython import parallel
        engines = parallel.Client()
```

Figure 2.5: Connecting to the IPython cluster.

2.2.5 Multicore programming with Direct View interface

In order to send command to the cluster, we need first create a Direct View interface to interact with all engines. Within the direct view, engines[0] represents the first engine, engines[1] the second engine, and so on.

```
In [2]: dview = engines[:]
```

Figure 2.6: Creating a direct view to all engines.

The command in figure 2.6 creates a Direct View interface to all engines. We give an example in figure 2.7 , this command is executed on the client and sends the command "a=2" to the first engine:

```
In [3]: engines[0].excute("a=2")
```

Figure 2.7: Sending command to engines.

We may retrieve the result by executing the following command on the client:

```
In [4]: engines[0].pull("a")
```

Figure 2.8: Retrieving result of execution.

With this, we end introducing the basic parallel computing commands.

2.3 Python and SQLite

Regarding the database. We have a database as the figure 2.9 shows. It is a file with file extension ".sqlite" and its size is almost 30GB. Therefore, we are going to introduce SQLite, which is the tool we use in order to manage our database.

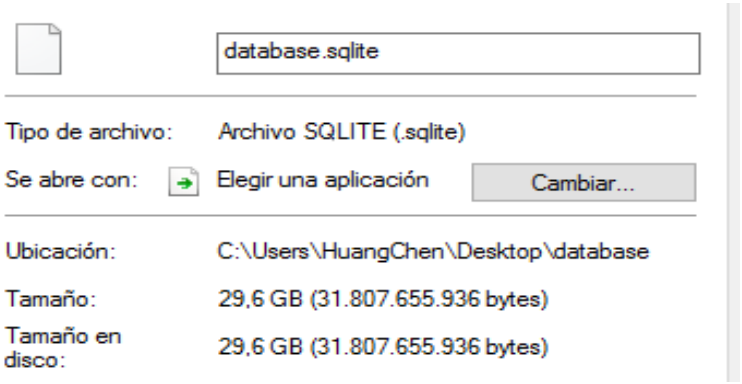


Figure 2.9: database file details.

SQLite is a C library that provides a lightweight disk-based database that does not require a separate server process and allows accessing the database using a nonstandard variant of the SQL query language. Some applications can use SQLite for internal data storage. It is also possible to prototype an application using SQLite and then port the code to a larger database such as PostgreSQL or Oracle.

In addition, the SQLite can be integrated with Python using sqlite3. The sqliste3 module provides a SQL interface compliant with the DB-API 2.0 which can suffice our requirement to work with SQLite database from our Python program.

To use the module, we must first create a Connection object that represents the database. Then, we can use the powerful Python data analysis toolkit Pandas to read it and convert it as a DataFrame tabular.

Pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with "relational" or "labelled" data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python.

DataFrame is a class inside the package Pandas. It is the primary Pandas data structure that has properties such as two-dimensional size-mutable; potentially heterogeneous tabular data structure with labelled axes (rows and columns); arithmetic operations align on both row and column labels; It can be thought of as a dictionary-like container for Series objects, see figure 2.10.

```
import sqlite3
import pandas as pd

sql_conn = sqlite3.connect('database.sqlite')

#Get a piece of data base from "database" file
def get_chunk(num):
    df = pd.read_sql("SELECT * FROM May2015 LIMIT "+num, sql_conn)
    return df
```

Figure 2.10: Reading data from database.sqlite file and convert it to DataFrame.

As the above figure shows, we first create a Connection object named "sql_conn" to represent our database. Then we use command "read_sql" to convert it into a DataFrame object named "df".

Once we get the converted database in a DataFrame object, we can now use Python command as usual to manage it.

Chapter 3

Design and implementation

The aim of this chapter is to explain the process followed in order to construct the table which contains the worst behaviour users in Reddit at May of 2015. It is divided into 5 sections: Extracting information from database, Behaviour analysis, Building histogram, Scheme, Adding Parallelism. The first section describes how we read information from database and what does those data mean. Behaviour analysis summarizes how we classify a Reddit user as a "bad" user. Building histogram presents the process to create the histogram by counting the occurrence of Reddit user. Scheme describes how we sorted list of "bad" users and how we got the worst users. Finally, Adding Parallelism section details how we added parallel computing in order to get the same result as the process we did at the previous sections.

3.1 Extracting information from database

3.1.1 Description of database

In order to extract data from database and get information of them. Firstly, we need a powerful and very useful tool to read data from a such big database which its size is 30GB. There are many other options, but we chose SqliteBrowser which is the most suggested one.

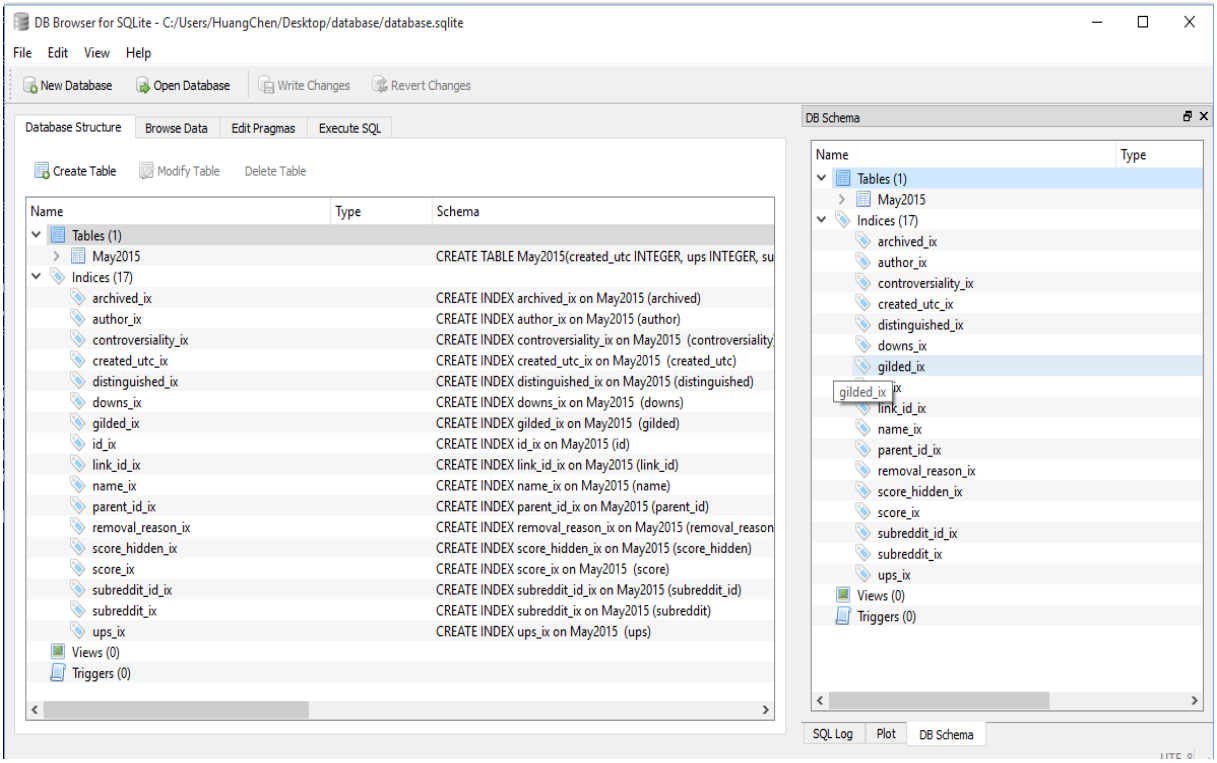


Figure 3.1: The interface of SqliteBrowser.

As the above interface (figure 3.1) shown, our database structure a table named May2015, which contains all the information about Reddit on May of 2015. Also, it has 17 indices that correspond 17 entities inside the table May, 2015. We detail it in the below figure.

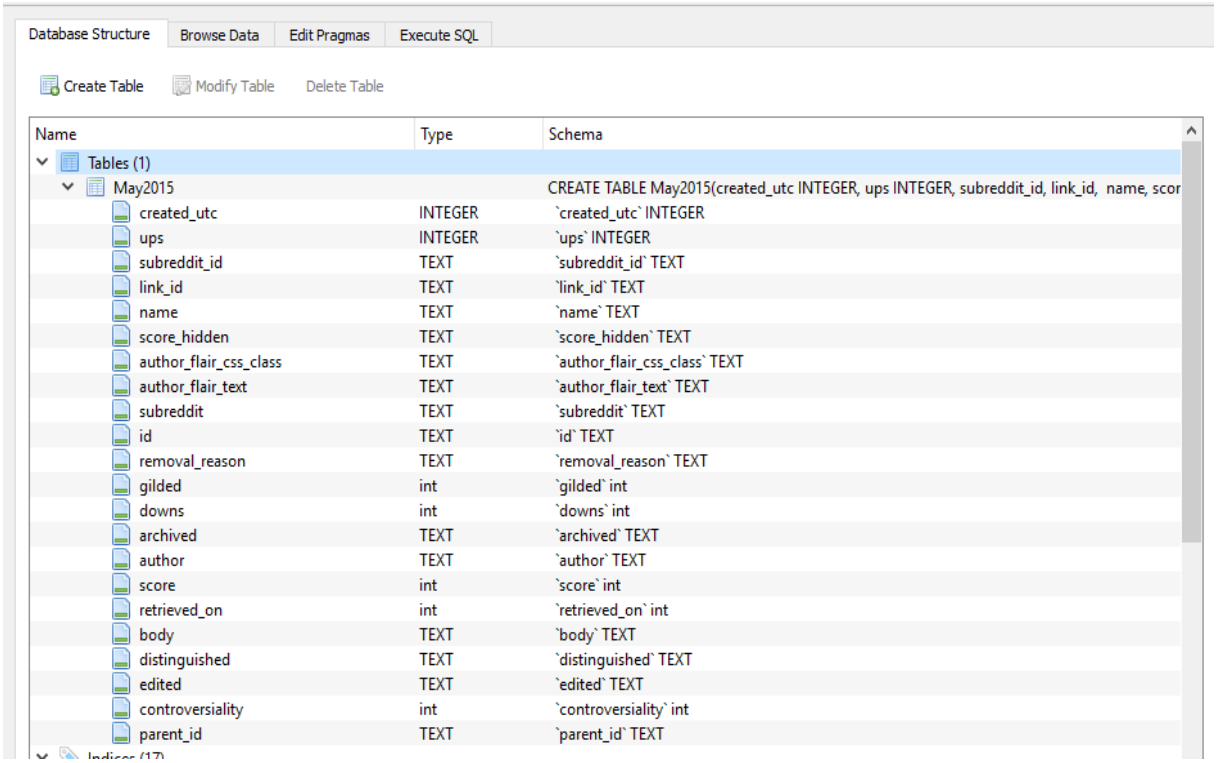


Figure 3.2: Expanding the table May2015.

With respect to the above expanded table, there are 22 rows in total which each row represents an attribute in database. That is, we can access to each attribute to get its domain. A domain of an attribute is the set of valid values for that attribute. For instance, the attribute "author" might have domain such as user name so that we can distinguish one to others. The first column shows the name of each attribute; the second column tells us what type those attributes belong to; the third column is just schema.

Even though there are 22 different attributes where we can get lots information, but we are going to access into just 2 of them which are "author" and "body". The reason is explained in the next section. However, we need to know some other 3 important attributes as well. They are these 5 below.

1. Subreddit: it is the name of subreddits in Reddit. As we mentioned before, each subreddit represents a topic in Reddit. There are 50 default subreddit topics such as movies, gaming, politics, relationships, fitness, food, history, etc. Also, there are many other subreddit topics created by redditors. For example: nba, cigars, eagles, etc.

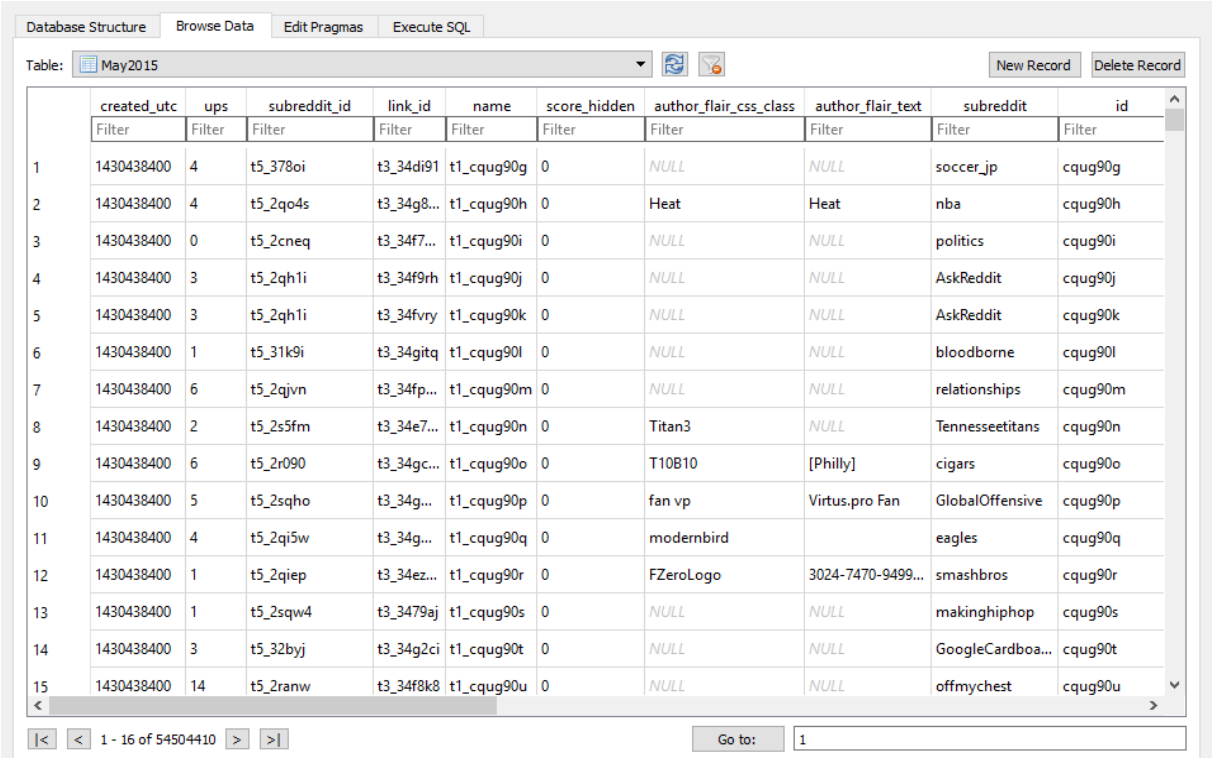
2. Subreddit_id: each subreddit has its own identity number which is a combination of a series characters and integer numbers.

3. Id: every comment or post made by a redditor has its own identity number. They are not repeated.

4. Author: it is the user name of redditors. Each redditor has they own user name. Also, the user name cannot be changed once it is registered in Reddit database.

5. Body: the content of post or comment written by redditors form a "body" in Reddit database. Its type is text.

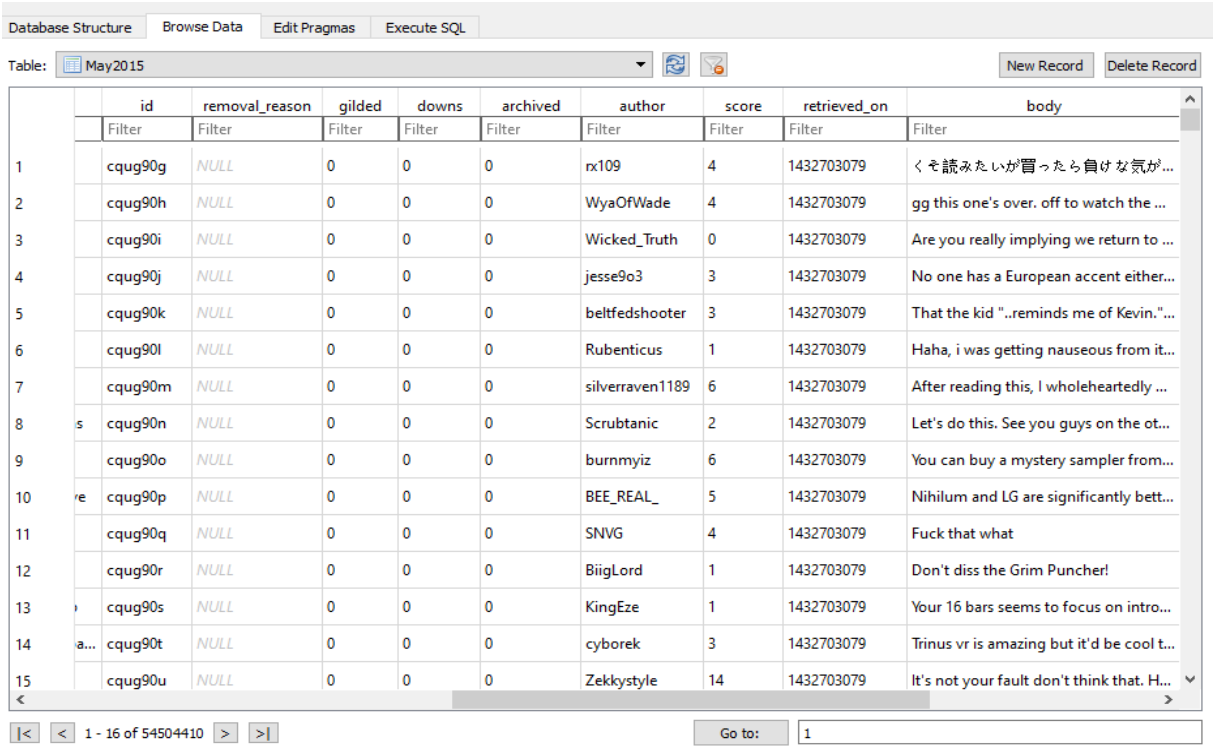
An entity is a set of objects of the same types that share the same properties or attributes, so it is characterized by an ordered list of attributes. In our case, an entity is a post or comment that has all the above 22 attributes. We can also use SqliteBrowser to browse data and see what are the real entities in Reddit database. The browsed data is shown on below two figures.



	created_utc	ups	subreddit_id	link_id	name	score_hidden	author_flair_css_class	author_flair_text	subreddit	id
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1430438400	4	t5_378oi	t3_34di91	t1_cqug90g	0	NULL	NULL	soccer_jp	cqug90g
2	1430438400	4	t5_2qo4s	t3_34g8...	t1_cqug90h	0	Heat	Heat	nba	cqug90h
3	1430438400	0	t5_2cneq	t3_34f7...	t1_cqug90i	0	NULL	NULL	politics	cqug90i
4	1430438400	3	t5_2qh1i	t3_34f9rh	t1_cqug90j	0	NULL	NULL	AskReddit	cqug90j
5	1430438400	3	t5_2qh1i	t3_34fvry	t1_cqug90k	0	NULL	NULL	AskReddit	cqug90k
6	1430438400	1	t5_31k9i	t3_34gitq	t1_cqug90l	0	NULL	NULL	bloodborne	cqug90l
7	1430438400	6	t5_2qjvn	t3_34fp...	t1_cqug90m	0	NULL	NULL	relationships	cqug90m
8	1430438400	2	t5_2s5fm	t3_34e7...	t1_cqug90n	0	Titan3	NULL	Tennesseetitans	cqug90n
9	1430438400	6	t5_2r090	t3_34gc...	t1_cqug90o	0	T10B10	[Philly]	cigars	cqug90o
10	1430438400	5	t5_2sqho	t3_34g...	t1_cqug90p	0	fan vp	Virtus.pro Fan	GlobalOffensive	cqug90p
11	1430438400	4	t5_2qi5w	t3_34g...	t1_cqug90q	0	modernbird		eagles	cqug90q
12	1430438400	1	t5_2qiep	t3_34ez...	t1_cqug90r	0	FZeroLogo	3024-7470-9499...	smashbros	cqug90r
13	1430438400	1	t5_2sqw4	t3_3479aj	t1_cqug90s	0	NULL	NULL	makinghiphop	cqug90s
14	1430438400	3	t5_32byj	t3_34g2ci	t1_cqug90t	0	NULL	NULL	GoogleCardboa...	cqug90t
15	1430438400	14	t5_2ranw	t3_34f8k8	t1_cqug90u	0	NULL	NULL	offmychest	cqug90u

Figure 3.3: Browsed data

As the figures 3.3 and 3.4 show, each row represents one entity, that is one post or comment. Each column represents the domain of the correspond attribute. In the bottom left



		id	removal_reason	gilded	downs	archived	author	score	retrieved_on	body
		Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1		cqug90g	NULL	0	0	0	rx109	4	1432703079	くそ読みたいが買ったら負けな気が...
2		cqug90h	NULL	0	0	0	WyaOfWade	4	1432703079	gg this one's over. off to watch the ...
3		cqug90i	NULL	0	0	0	Wicked_Truth	0	1432703079	Are you really implying we return to ...
4		cqug90j	NULL	0	0	0	jesse9o3	3	1432703079	No one has a European accent either...
5		cqug90k	NULL	0	0	0	beltfedshooter	3	1432703079	That the kid "..reminds me of Kevin."...
6		cqug90l	NULL	0	0	0	Rubenticus	1	1432703079	Haha, i was getting nauseous from it...
7		cqug90m	NULL	0	0	0	silverraven1189	6	1432703079	After reading this, I wholeheartedly ...
8	s	cqug90n	NULL	0	0	0	Scrubtanic	2	1432703079	Let's do this. See you guys on the ot...
9		cqug90o	NULL	0	0	0	burnmyiz	6	1432703079	You can buy a mystery sampler from...
10	re	cqug90p	NULL	0	0	0	BEE_REAL_	5	1432703079	Nihilum and LG are significantly bett...
11		cqug90q	NULL	0	0	0	SNVG	4	1432703079	Fuck that what
12		cqug90r	NULL	0	0	0	BiigLord	1	1432703079	Don't diss the Grim Puncher!
13	)	cqug90s	NULL	0	0	0	KingEze	1	1432703079	Your 16 bars seems to focus on intro...
14	a...	cqug90t	NULL	0	0	0	cyborek	3	1432703079	Trinus vr is amazing but it'd be cool t...
15		cqug90u	NULL	0	0	0	Zekkystyle	14	1432703079	It's not your fault don't think that. H...

Figure 3.4: Browsed data

corner, it tells us that we have in total 54.504.410 entities which are posts and comments made by redditors during the whole month of May in 2015.

After we browsed the data, we have to extract them and use them to analyse users' behaviour so that we could get our goal.

3.1.2 Extracting data

We have already seen in the chapter 2 that we firstly use SQLite to read data, then we can store them in a DataFrame tabular. After that, we can use the DataFrame object to process our database. For instance, we can visualize clearly the first 4 entities of data from database as a DataFrame tabular.

```
In [14]: df[:4]
Out[14]:
```

	created_utc	ups	subreddit_id	link_id	name	score_hidden	author_flair_css_class	author_flair_text	subreddit	id	...	downs	a
0	1430438400	4	t5_2qo4s	t3_34g8mx	t1_cqug90h	0	Heat	Heat	nba	cqug90h	...	0	(
1	1430438400	0	t5_2cneq	t3_34f7mc	t1_cqug90i	0	None	None	politics	cqug90i	...	0	(
2	1430438400	3	t5_2qh1i	t3_34f9rh	t1_cqug90j	0	None	None	AskReddit	cqug90j	...	0	(
3	1430438400	3	t5_2qh1i	t3_34fvry	t1_cqug90k	0	None	None	AskReddit	cqug90k	...	0	(

4 rows × 22 columns

Figure 3.5: First 4 entities visualized by DataFrame tabular

Furthermore, we can powerfully manage the database using DataFrame, include analysing user behaviour by process the domains of attribute "body". This leads us to expand the next section topic.

3.2 Behaviour analysis

In this section, we are going to discuss how we analyse Reddit user's posts and in which manner we evaluate the "bad" rate to each redditor.

Firstly, as we mentioned at the previous section, even though we have 22 different attributes, we just need to use the attributes "author" and "body". That is because each redditor has his own user name. Although, in very few cases does Reddit exist some repeated account. But, the repeated account's amount is negligible so that we can consider that all the user name of redditors are unique. In addition, due to the huge size of database, we should firstly just process the 1.000.000 first entities in order to save runtime.

Thus, we aim to analyse the posts and comments made by redditor and to find out the one who behaviours worse in Reddit. Therefore, we need a list of insult words so that we can check if the comments contain insult words in that list. That is, to create a list of bad words, the more insult words it contains more accurate for analysis result. Hence, we have a text file named "badlist.txt"[1] (based on [3]) which contains in total 496 insult words. The words in that file are also considered network language problem. For example, in order to control network language violence, some social network sites do not allow people post insult content. Thus, people insult the word "shit" as "sh*t" so that we could not find it easily. That is the reason we have such lots words in the file. Because we add the hottest insult words with some of its possible network language manner such as the word "fuck", its possible network language could be "f u c k", "f u * k", etc.

After get the file, we need to read line by line in order to store all these insult words into a Python variable named "badwords". The type of "badwords" is Python list.

Before we detail more, I need to explain first the criterion we are going to use in order to calculate the "bad" rate for each redditor.

3.2.1 Bad rate BR_P

We mentioned the "bad" rate before. It is an integer number we need to calculate for each redditor by analysing all the comments they posted in Reddit during May of 2015.

The criterion to calculate the redditor bad rate is the follow equation:

$$BR\_P = max(\ \tfrac{A}{B})\cdot 100\cdot 30\% + \tfrac{C}{D}\cdot 100\cdot 30\% + \tfrac{E}{F}\cdot 100\cdot 40\% \tag{3.1}$$

We write the "bad" rate as BR_P. From the above equation, we can get that the minimum value of BR_P is 0 and the maximum is 100, because we multiply 100 in each term on the right side hand.

We now have to describe the meaning of each of characters A, B, C, D, E, F on the above equation. All of them are integer numbers. The meaning of them are the following:

[1] The file is stored together with source code

- A: In one comment, A is the number of total insult words in that comment.
- B: It is the total comment words that contains A insult words.

So, $\frac{A}{B} \cdot 100$ is the proportion of insult word in one comment. But, we multiply $max(\frac{A}{B}) \cdot 100$ instead of $\frac{A}{B} \cdot 100$. Because, in this first term we evaluate the worst comment that redditor have ever written.

- C: The total number of insult words written by one redditor.
- D: The total number of comment words written by one redditor.

Hence, $\frac{C}{D} \cdot 100$ is the proportion of insult word written by one redditor.

- E: The total amount of insult comments written by one redditor. An insult comment is the comment which contains at least one insult word.
- F: The total amount of comments posted by one redditor.

Therefore, $\frac{E}{F} \cdot 100$ is the proportion of insult comment posted by one redditor.

We can see the following example to make the reader clearly understand all the significance of the above characters and equation.
Assume that a redditor with user name "hey_reddit" posted in total 4 comments in May of 2015. These 4 comments are below:

1. I just wonder how the fuck do you write so fast, fuck .
2. King is very technically talented but hi throws a lot of shit at the wall and not all of it sticks.
3. I've personally never been able to stick with a King book for long because I can't stand his writing, but Martin's not much better for a lot of the same reasons.
4. I always thought he was horribly overrated and that his stories generally make better movies than written media.

From the above 4 comments, we can find easily that the first 2 comments have bad words that are contained inside the file "badlist.txt"[2]. So, the comment 1 has two insult word which are fuck, fuck. The comment 2 has also one bad word which is shit. The rest 2 comments have zero bad word.

- Hence, in comment 1, it has the value of A equals 2. It has in total 12 words so that B equals 12. Then $\frac{A}{B}$=0.182
- In case of comment 2, it has A=1, B=21. So, $\frac{A}{B}$=0.048
- In case of comment 3, A=0, B=31, $\frac{A}{B}$=0.
- In case of Comment 4, A=0, B=18, $\frac{A}{B}$=0.

[2]The file is stored together with source code

So, the maximum value of $\frac{A}{B}$ is 0.182.

Now, by definition of the character C, the redditor "hey_reddit" has in total 2 insults. So C=3, D=12+21+31+18=82 and $\frac{C}{D}$=0.037 Also, E=2 because there are just 2 comments that contain bad words, and F=4 because he posted in total 4 comments during whole month of May. Then, $\frac{E}{F}$=0.5

After calculating all the values of A, B, C, D, E, F. We can now get the bad rate BR_P to the redditor "hey_reddit".

$$BR\_P = 0.182 \cdot 100 \cdot 30\% + 0.037 \cdot 100 \cdot 30\% + 0.5 \cdot 100 \cdot 40\% = 26.57 \tag{3.2}$$

So, this example ends with BR_P=26.57 which is the "bad" rate obtained to "hey_reddit".

In spite of the obtained value of BR_P, we have also to explain the reason of why we chose these terms max($\frac{A}{B}$), $\frac{C}{D}$, $\frac{E}{F}$ and their corresponded weight value 30%, 30% and 40% in each term. We set a weight value of 30% in term of max($\frac{A}{B}$) because regarding to one comment, we consider that the comment is written very rudely if it has lots insult words. So, we take the maximum value of all values of $\frac{A}{B}$, that is to value the roughest comment posted by a redditor. Then, we also have to take into account of the amount of total insult words written by that redditor. Because maybe that redditor has very high value in term of max($\frac{A}{B}$) due to his personal reason during the time he wrote the rudely comment, but he does not usually insult, so we set a weight value of 30% in term of $\frac{C}{D}$ to make his BR_P decay. Finally, we set the highest weight value to the term of $\frac{E}{F}$, because we emphasize much more on the frequency of redditors insult.

However, the above equation calculates just the provisional "bad" rate, that is the reason we write it as BR_P. Now, we have to discuss another approach which is the equation to calculate the final "bad" rate of each redditor.

3.2.2 Bad rate BR_F

What the previous equation lacks is that it does not consider the problem of difference between active and inactive Reddit user. In other words, an active user is the redditor which frequently posts or comments in Reddit. We should focus on analysing behaviour of active users, but that does not mean we should ignore those less active user's comments.

There are totally 3 different classes to categorize Reddit users in an active degree. Each level of active users in Reddit is classified by the monthly number of users' appearance in Reddit.

1. Low active users: the redditor whose appearance is less than 3 in the first 1000000 Reddit posts.

2. Normal active users: the redditor who has appearance between 3 to 6.

3. High active users: the redditor whose appearance is more than 6.

The reason of why is the classification made by these number (3, 6) will be explained in the next section 3.3.

The above classification we made by processing just the first 1.000.000 entities due to computational restriction. But, the number of total monthly comments in Reddit is 54.504.410. Hence, we calculate the proportion 3 : 1.000.000 to 54.504.410 . Then, we get the following correspondent users' appearance to all 54.504.410 comments in the database.

1. Low active users: The Reddit user who appears less than 30 times per month, that is the redditor who posts less than 30 comments in May of 2015.

2. Medium active users: The Reddit user who has between 30 to 60 posts per month in Reddit.

3. High active users: The Reddit user who has more than 60 posts per month in Reddit.

Now, we need to give a weight value to each class of active users. The weight value is a real number between 0 to 1 which represents the level we attach importance to the class users' posts.

We claim that high active users in Reddit are the targets we are going to analyse on. Hence, we give a weight value of 100% to them. Then, the medium active users have the weight value equals 70%. Finally, we consider that the comments posted by low active users are less important than the others. The reason is, if a redditor just posted once in May of 2015, and that comment just contains one word which is a bad word. Then, that redditor must have BR_P=100 so that it would be considered the worst behaviour user. Although, that person is theoretically the worst behaviour user, but it is not what we are interested on. Therefore, we set a weight value of 30% to less active user in order to decay their "bad" rate value and avoid the above case.

The new equation to calculate the final "bad" rate is below:

- In case of low active user, then $BR\_F = BR\_P \cdot 0.35$

- In case of medium active user, then $BR\_F = BR\_P \cdot 0.70$

- In case of high active user, then $BR\_F = BR\_P \cdot 1.00$

3.2.3 Algorithm with flowchart

After we clarify how we calculate the "bad" rate for each redditor, we need to detail more about how to analyse user behaviour by using "bad" rate, what the workflow is.

Thus, in figure 3.6 the diagram shows each step of the whole process using top-down and left-right approaches.

As the flowchart (figure 3.6) shows, we divide the program into various steps.

1. The first step we have to do is to import the database file and convert it into a DataFrame tabular. We have already seen this step in the section of Extracting data. Once we obtained the data stored in the DataFrame tabular named "df", we need firstly reform the DataFrame tabular. That is, removing the rest contents of entities except "author" and "body".

2. Then, we have to create an empty Python dictionary named "dic" in order to store the process result.

3. The next step is to read the first entity in "df". Then, we get the domain of attribute body which is the content of comment. We split the comment text into a list of words by spaces between words. Then we check each word in the list whether that word is also contained in the python list "badwords" (We mentioned it in the initial of this section). If there is no word match to "badwords" list, it means that comment does not contain any insult word and we are not going to do anything with it, then we back to the previous step in order to read the next entity's content. Otherwise, the comment has at least one insult word so that we need to do the next step.

4. In this step, we firstly count how many insult words the comment have, then we calculate the value of A, B, C, D, E, F which are the characters we need to calculate BR_P. Then, verify if the "author" that the insult comment belongs to is already exist in the dictionary "dic". If not, we have to create a new item such as

   {author: {"C":0,"D":0,"E":0,"F":0, "A/B":0,"body": df.body[i] }}.

   (a) The key is the redditor's user name.

   (b) The value is again a Python dictionary with 6 items. The first 4 items are obvious. In the fifth item, the key is a string "A/B", its value is the maximum value of $\frac{A}{B}$ comparing all the rest insult comment written by this redditor. The last item is actually the content of comment which its $\frac{A}{B}$ value is max.

   In another case, we just need to refresh the value of "C", "D", "E", "F", "A/B" and "body", then we update the dictionary "dic".
   No matter in which of the above 2 cases, we finish process one comment in this step. Therefore, in both case follow the same below step.

5. As we have finished processing one entity, we now need to check if it is the last entity in "df". If not, we back to the step 3 in order to process the next comment until all the entities are processed. Otherwise, we go toward to the next step.

Figure 3.6: The flowchart of whole program.

6. After run all the above steps, we get the Python dictionary "dic" which contains all the redditors that insult in Reddit and many other information. Now, we add a new item to the dictionary which classify the active degree of each redditors in "dic". The key of this new item is "Weight" which is a string value. Its value is a Python float number, it could be 0.35, 0.70 or 1.00. Thus, an example of the updated dictionary "dic" is below: {author: {"C":0,"D":0,"E":0,"F":0, "A/B":0,"body": df.body[i],

"Weight":0.70}}.

7. We calculate the BR_F value in this step. Then, we create a new dictionary item to store the value in order to add it to the dictionary "dic". The new item structure as {"BR_F": float_number}. The value of this item is BR_F value. So, it is a float type number between 0 to 100.

8. Finally, we sort the result dictionary "dic" by "BR_F" in descending order. Then visualize the highest value users in a DataFrame tabular.

The section ends with the above diagram and the algorithm details.

3.3 Building histogram

In this section, we discuss what implementation we did in order to categorize the degree of active users, which we mentioned at the previous section and use it to calculate BR_F. Moreover, we also show some histograms to explain the implementation result.

In order to do classification, we need to create a Python program to count the appearance of each redittor and build the histograms to set the classification value (the weight value). First of all, for the same reason we talked before, we just process the first 1.000.000 entities in the database.

1. The first step is, extracting data (the first 1.000.000 entities) from the database and store it as a DataFrame tabular. Then, we create a list of redditors named "author_list" which appear in those entities. In this list, there is no repeated redditors and the total number of redditors is 322.216.

2. In "author_list", we count each redditor's appearance in the first 1.000.000 comments in our database.

3. Convert it as a Python dictionary so that each key is a Reddit user name, and the value is the correspond redditor's appearance.

4. Finally, we build histograms of the result dictionary to visualize them and store them as picture files. In addition, we also show the result in a DataFrame tabular with numbers, whilst histogram describes result data with diagram. The result in the DataFrame tabular is stored in an Excel file as well. The name of the Excel files is "Appearance_and_active_author(1000000_comments).xlsx"[3].

The detail of programming code is attached in the IPython notebook file named "Histogram_about_1rst_1000000_comments.ipynb"[4].

The above algorithm details the implementation of building histogram in order to categorize the degree of active Reddit users. Here, we are going to show the result histograms

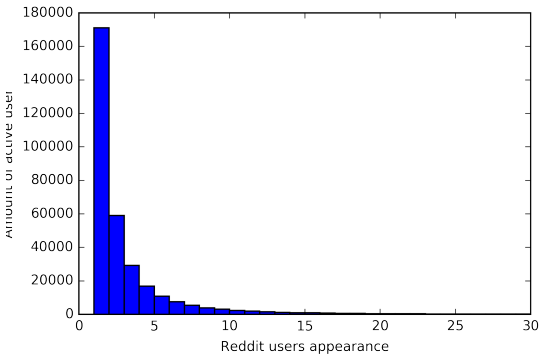


Figure 3.7: Appearance 0-29

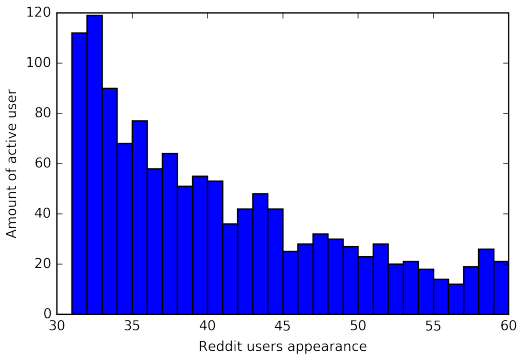


Figure 3.8: Appearance 30-59

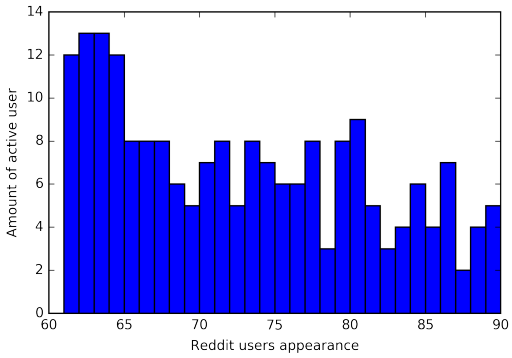


Figure 3.9: Appearance 60-89

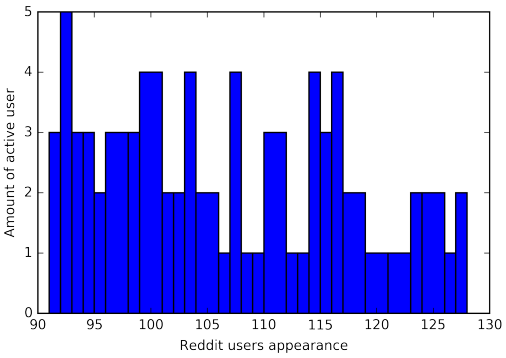


Figure 3.10: Appearance 90-129

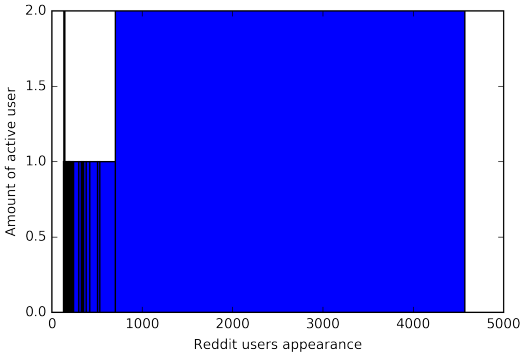


Figure 3.11: Appearance 1-5000

and discuss how to use this result in furtherance of classifying active redditors.

In figures 3.7, 3.8, 3.9, 3.10, 3.11, all these 5 histograms, the x-axis represent the appearance of redditors, the y-axis is the amount of redditors with correspond appearance in x-axis. For example, in the first histogram, the first blue bar represents that between 160.000 to 180.000 redditors posted 1 comment in Reedit. Likewise, the first figure shows the proportion of the amount of redditor with appearance between 1 to 29 during the first 1.000.000 comments in May of 2015; the second figure shows the appearance between 30 to 59; the third one, 60 to 89; the four one, 90 to 1000. The firth, the rest until 5.000.

The total amount of comments we processed is 1.000.000, but there are just 322.216 active users. That means, these 322.216 users posted the first 1.000.000 comments in Reddit at May,2015. Then, $\frac{1.000.000}{322.216} = 3.104$, we can theoretically conclude that one user posted 3 comments is a medium active user. Moreover, we can see that there is a big gap between the first 2 bars to the rest bars by the first histogram. The first 2 bar have y-axis value higher than 30000, diversely the y-axis value are lower than 30000 since the third bar. Hence, consider the above two reasons, we classify the low active users are those who has appearance less the 3.

In case of medium active users, we classify them as the redditors whose appearance is between 3 to 6. That is because in the first histogram, the third bar to sixth bar have y-axis value between 7.500 to 30.000. Then, from the seventh bar, its y-axis value down to 5.500 and below. Therefore, we consider that the high active users are those redditors whose appearance is higher than 7. In addition, we can clearly say the above number thanks to the Excel file which we mentioned in the previous section, that is a part of histogram we also stored the result in an Excel file. The first 10 position in the attached Excel file is shown in the figure 3.12:

In fact, we also did just one histogram with the appearance between 0 to 5000, but the histogram shows unclearly and tidy bars due to its big range. That is the reason we split the result into 5 histograms. With respect to the above histograms and the screen shot of the Excel file, we discussed how we classify the degree of active Reddit users with their appearance.

3.4 Scheme

In fact, there exists many other approaches to perform this project. We have been talking in all the above two section about using DataFrame from Python package Pandas, but actually we combined both of DataFrame and Counter to manage data.

Collections module is one of the Python standard library that emphasize on its high performance container datatypes. This module implements specialized container datatypes

[3]The file is stored together with source code
[4]The file is stored together with source code

	A	B	C
1		Appearance	Amount of active author
2	1	1	171093
3	2	2	59065
4	3	3	29150
5	4	4	16917
6	5	5	10854
7	6	6	7576
8	7	7	5445
9	8	8	3881
10	9	9	3140
11	10	10	2256

Figure 3.12: The screen shot of the file "Appearance_and_actived_author(1000000_comments).xlsx".

providing alternatives to Python's general purpose built-in containers. We are going to focus on one of its class, Counter.

Counter is a dictionary subclass for counting hashable objects. It is an unordered collection where elements are stored as dictionary keys and their counts are stored as dictionary values. A counter tool is provided to support convenient and rapid tallies.

Actually, we can use Counter to perform the whole project. But, in step of result data management. Due to small size of result data, DataFrame provides us user-friendly command to use and clear visualization of result.

Likewise, we did not use DataFrame in whole program in account of its slow runtime. It does not mean DataFrame is slow in execution time, but we found that it is slower then Counter after we did various tests.

Therefore, we appropriately combine these two useful data analysis tool to perform the project. The another main reason that Counter is not being deprecated because of its useful collecting data tool in performing parallel computing. That is what we are interested in discussing at the next section.

3.5 Adding Parallelism

The reason of performing project with parallel computing is introduced in the Chapter 2. Hence, we are going to detail the implementation of adding parallelism to the previous

programs.

3.5.1 Algorithm of behaviour analysis with parallel computing

Assume that the database file does not fit into memory. Therefore, the client (i.e. the procedure) split the input data into uniform chunks of appropriate size. In this case, we set the size of each chunks as 20 entities, so 20 Reddit comments (The reason of 20 is based on the section Results in the [6]). The whole implementation is available as a IPython notebook. We discuss here only those issues related to parallelization issues.

First, let direct view be a IPython object associated to all the engines of the cluster. We set the block attribute to True, that is, by default all the commands that are sent to the engines will not return until they are finished. In order to be able to send tasks to the engines in a round robin-like behaviour, an infinite iterator over the list of engines can be created. This can be done with a Cycle object:

```
In [1]: from itertools import cycle
        c_engines = cycle(engines.ids)
```

Figure 3.13: cycling engines

We split the workflow that we saw in section 3.2 into 2 different parts in account of adding parallelism. Both parts share the same flowchart (figure 3.14):

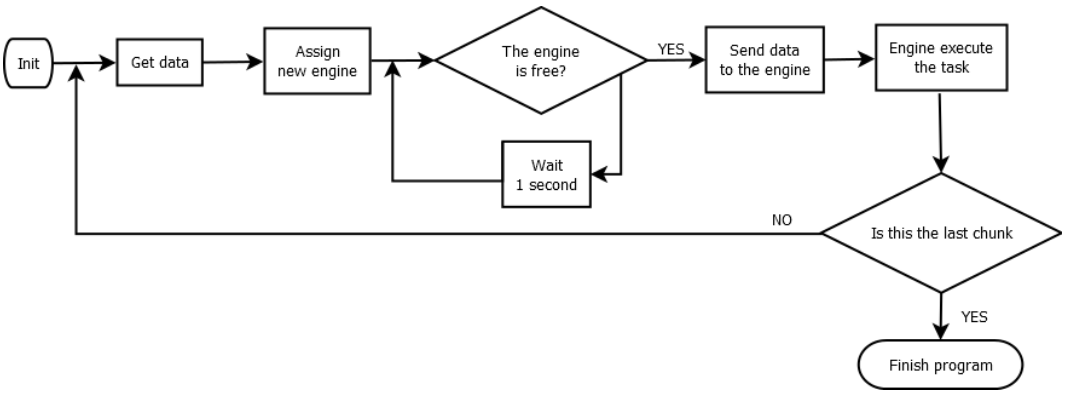


Figure 3.14: Flowchart to analyse Reddit user behaviour adding parallelism.

In the first part, the task that is executed in engines, it actually takes the first 5 steps of the algorithm in section 3.2 and then add to it the above parallel compute. Likewise, in the second part we add parallelism to the rest 3 steps of that algorithm. In fact, the result of executing the first part gives us the list of redditors with their correspond BR_P

value. The second part gives us the final result, that is the sorted list of redditors with their correspond BR_F. In the following, we explain each step of the above diagram.

1. We begin by sending to each engine all the necessary functions that are needed to process the data. Among these functions we just may mention 2 funtions. The first one is the initialization function, which resets local engine's variables. The second one is the main function, which applies process a chunk of lines and groups results in a variable, which is local to each engine. After sending the necessary functions to the engines we execute in each engine the initialization function, in order to initialize the local variables in each engine.
   The execution is performed in non-blocking mode. That is, the initialization function is executed on each engine without waiting for the engine to finish. Thus, the loop can be executed for each engine in parallel.

2. The client reads a chunk of the file and selects to which engine the chunk will be sent. Each read chunk will have the same number of lines with the exception of the last chunk read from the file, and thus the runtime in each engine we expect each chunk to be processed is the same.

3. Once the chunk has been read and the engine that will process the chunk has been selected we need to wait for the engine to finish its previous task. While the engine has not finished, we wait for 1 second then check the state again. If still not, we wait for 1 second, check and so on. We do it until the engine is free.

4. At this point we are sure that the current running engine is free. Thus, we may send the data to the engine and ask it to process it.

5. In this first part, the task we ask the engine to process is executing the main function to calculate BR_P of each redditor. Then the main function locally aggregates the result of analysing each chunk in that local variable. At the end the client will collect the local results of all engines.

6. We now check if the chunk that the engine has just processed is the last chunk. If not, the algorithm then jumps again to step 2. Otherwise, we go to the next step which is the last step.

7. Once the loop (steps 2 to 5) has processed all the chunks in the file, the client gets the results from each engine and aggregates them into a new variable which is the Python dictionary type. After reading all the results from the engines, the final result is stored in the dictionary that we have just mentioned.

With the above algorithm, we get a dictionary of the execution results which its keys are list of redditors, the value of each key is again a dictionary. The item in the value dictionary store the correspond BR_P and the information to calculate BR_P.

Now we need use again the above algorithm with some changes in the step 1 and 5 in order to perform the second part of project which is to calculate BR_F value. That is, in step 1 instead of sending whole database to engine, we send the above dictionary of

results. Then, in step 5, we execute the function to assign the correspond weight to each redditor by counting its appearance. Also we calculate the BR_F value in step 5 whilst the first part it calculates BR_P.

Once run twice the above algorithm, we get the result store in a dictionary which contain the redditors that insult in Reddit and its correspond "bad" rate BR_F. The last step is to convert it into a DataFrame tabular and show the result. Moreover, we store it as an Excel file named "final_result_top_20.xlsx"[5].

[5]The file is stored together with source code

Chapter 4

Result

The purpose of this chapter is to present the results of this project. It is divided in two sections: Behaviour analysis, Comparison of Parallelism and Non-Parallelism computing. Both of them detail the result of what we did in the previous chapter showing the graphic either run chart, mutable file or Excel files.

4.1 Behaviour analysis

The result of our goal in this project is presented in this section. We show the result in a DataFram tabular inside the IPython notebook, also we store it in an Excel file "Final_result_top_20.xlsx"[1]. This file stores the final result of behaviour analysis with parallel computing (see section 3.5), which is the 20 worst behaviour users in May of 2015. As it is the result of our aim in this project, hence we show it in figure 4.1.

The first column of the result table (figure 4.1) shows the user name of these "bad" behaviour users.

The second column with the column name body represents the worst comment that redditor posted in May of 2015. In other words, the comments which has higher value of $\frac{A}{B}$.

The column 3, 4, 5 contain the value of $max(\frac{A}{B} \cdot 100)$, $\frac{C}{D} \cdot 100$, $\frac{E}{F} \cdot 100$ respectively.

The column "Weight" describes the appearance or the active level of the correspond row's redditor. Remember that the high active users have the weight value 1. The normal active users have the weight value 0.70 and the low active users 0.35.

The last column "BR_F" shows the BR_F value (the bad rate) of the redditor in the same row level. The higher is the BR_F value, the worse is that redditor commented in May of 2015.

Now, we focus on the first row, which contains the information of a comment written by the redditor with Reddit user name "i_blue_my_self". That comment is the worst comment "i_blue_my_self" has ever had, and it just has one word which is "Cock". Then, in the

[1]The file is stored together with source code

	Author	body	(A/B)%	(C/D)%	(E/F)%	Weight	BR_F
0	i_blue_my_self	Cock	1	1	1	0,7	70
1	BrokenAssEnglish	Pussy!	1	0,066116	0,7	1	60,1
2	CSGOHT	Idiot	1	0,039841	0,636364	1	56,8
3	Taeyyy	Let's fix the ratio VAGINA VAGINA	0,9375	0,638298	0,166667	1	54,125
4	365brah	lmfao	1	0,019763	0,571429	1	53,4
5	gb12rulez	Shit	1	0,111111	0,5	1	53,3
6	wormriderdeluxe	Jingle All The Space Balls	0,2	0,2	1	1	52
7	F22xRaptor	COX!	1	0,025	0,5	1	50,9
8	elpollofrito	Damn	1	0,208333	0,357143	1	50,7
9	dIGITAL_cLARKE	Dildos	1	0,078947	1	0,7	50,68
10	Jackdude01	damn	1	0,014706	0,5	1	50,3
11	sgtstraps	hey scrub suck my cock. get slappe	0,2	0,033195	1	1	46,9
12	thundercatjones	Boner	1	0,111111	0,333333	1	46,5
13	derpeddit	Dildo.	1	0,059701	0,363636	1	46,2
14	Mr_Wyld	Damn	1	0,055556	0,352941	1	45,8
15	jimbovt	nice cock	0,5	0,106061	0,666667	1	45,1
16	biggestnerd	Damn	1	0,012162	0,368421	1	45,1
17	mr-president95	Rape rape rape, rape. Patriarchy, r	0,857143	0,069149	0,416667	1	44,61429
18	selfproclaimed1	Damn	1	0,074324	0,3125	1	44,5
19	BlueInventive	Masturbate.	1	0,107143	0,272727	1	44,1

Figure 4.1: the 20 worst behaviour users.

column "(A/B)%", its value of $max(\frac{A}{B} \cdot 100)$ equals 1, this means the worst comment that redditor wrote has the whole comment of insult words. In fact, that is obvious because the worst comment is shown in the column "body". Then, in the column "(C/D)%", it has value 1, hence all the words commented by that redditor are insult words. Now, it also has value 1 in the column "(E/F)%", this shows that every comment this redditor wrote in Reddit was insult comment. Looking at "Weight" column, its value is 0.70 so that this author is a medium active Reddit user. Finally, this redditor gets 70 points for his BR_F value which is the highest one comparing to other redditors. That is, this redditor is the worst behaviour user in Reddit evaluated by the BR_F value.

Moreover, we look at the second row which is the second highest BR_F value user. This user has $max(\frac{A}{B} \cdot 100)$ equal to 1 which we can easily see in the column "body" . Then, its $\frac{C}{D} \cdot 100$ has value 0.066116 so that he wrote few insult words in his comment. However, 70% of his comments contain insult word, this is shown in the column "(E/F)%". The weight value equals 1, it tells us this redditor is a high active redditor. Lastly, its BR_F value is 60.1 which is calculated by the equation we saw in the chapter 3. Likewise, we can intuitively explain all the information contained in the figure 4.1.

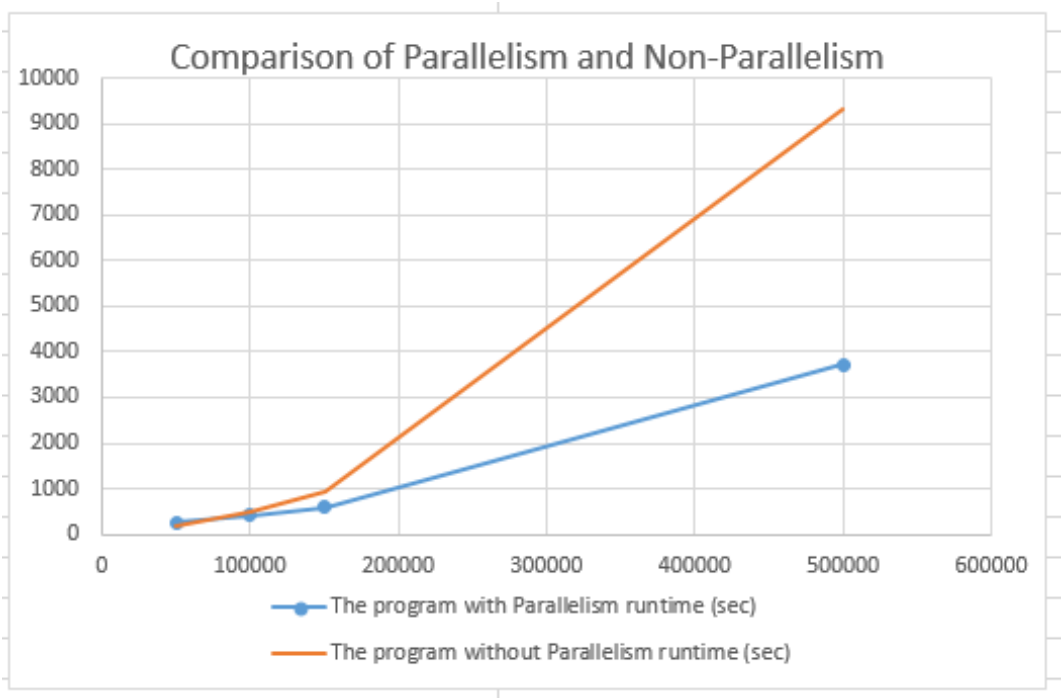


Figure 4.2: Run chart of the comparison of Parallelism and Non-Parallelism

4.2 Comparison of Parallelism and Non-Parallelism computing

We repeated various time before, we aim to raise the execution speed so that to save runtime. Thus we added parallel computing the program in the section 3.2. In this section, we are going to show that the execution speed is indeed raised. Before that, we need to mention that the execution is done in a laptop with 8GB of memory RAM. Its CPU is Intel Core i7-4510U which has 4 cores.

The result of the comparison between executions of the program with and without Parallelism is shown in the figure 4.2. Firstly, the x-axis is the amount of processed comments. Hence, we can easily see that we executed both programs processing 50.000, 100.000, 150.000 and 500.000 comments. Secondly, the y-axis represents the correspond runtime to the amount of comments processed in x-axis, the runtime is presented in the unit of seconds. Moreover, the blue line shows the runtime of executing the program with parallel computing. In like manner, the orange line represents the program with non-parallel computing.

Now, looking to the numbers. Both lines start at the x-axis equal to 50.000 which is the first blue point. This is the result of processing 5.000 comments comparing the execution result between parallel and non-parallel computing programs. Here, processing 50.000 comments takes around 253 seconds. Contrastively, the runtime of processing

50.000 comments with non-parallel computing program is 163.4 seconds, which is less the program with parallelism. This is due to small size of the taken data.

In case of analysing 100.000 comments of data, the program with parallel computing takes 415.8 seconds to perform it whereas the program without parallelism need 493.3 seconds. The difference is just around 1 minute.

Now we process 150.000 comments. The strength of parallel computing is obvious. The runtime of the Parallelism program is 601.39 seconds whilst the another one is 959.29 seconds. Comparing them, using the program with parallel computing save almost 360 seconds in runtime.

Lastly, the program with parallelism used 3719.50 seconds to process 500.000 comments whilst non-parallel computing program need 9316.46 seconds.

Hence, we conclude that maybe using parallel computing is not prominent in processing small database. But as the figure 4.2 shows, the more data size rises, more outstanding the parallelism is.

Chapter 5

Conclusion

This last chapter details the conclusions of this project and what further work we can propose to do it.

5.1 Degree Final Project conclusion

Based on the results at the previous chapter, wo can conclude the project as the following aspects.

We recall the main goal of this project is to analyse a database and find out the worst behaviour Reddit users. However, we did extract data from database; use the algorithm to analyse users' behaviour; calculate the BR_F value to every user who at least has insulted once in Reddit. With all of these mentioned steps we did, we get a list of redditors with high value of BR_F. Thus, these redditors are supposed to be bad behaviour users. Nonetheless, that does not mean the rest of redditor are good users. Because, this project works to calclute BR_F which is to combine some factors in order to evaluate users' behaviour as bad user. These factors are the user's worst comment, the amount of insult words commented, the user's appeareance. Therefore, we did not take in account of analyse other factors to differentiate a redditor as a "good" Reddit user or "bad" Reddit user.

Regarding to multicore parallel computing, comparing to the whole Reddit database or other business database, our database is small. But they usually use a supercomputer to manage the database, we use a laptop with just quad-core processor so that it is still big database. Many problems we have encountered during this project came from parallelism part. One of these problems was, the program worked well until we add multicore parallel computing. That was actually we did not send data to engine well, or during the processing of data we did not collect data well. But, even finding all these problems was so hard due to the interacting of the scheduler and engines is hide from users. Hence, solving problems was also hard, but performing project with parallel computing is still fascinated part. In addition, the previous chapter shows the improvement of runtime by

using parallel computing will be huge as we increase the database's size.

Lastly, we processed just the first 1.000.000 entities whilst the number of total entities is 54.504.410 in the database. Despite the working laptop has a quad-core processor and we process data with parallel computing, but it still takes almost 5 hours and half to process just 1.000.000 entities, which is less the $\frac{1}{5}$ of the total amount of entities. Regarding to the above reason, we tested the functionality of the algorithm with just a piece of database. Nevertheless, the algorithm we discussed in the previous section is not going to be affected by increasing database's size. The only difference is that to process whole database takes longer time to archive the goal. However, the final result would be changed if we process whole database. Thus, we consider that the main goal of project is fulfilled.

5.2 Feedback relevance

In order to perform this project, there are many other approaches. The algorithm we have seen in the section (3.2) is just one of the possible approaches. For example, the criterion we used to calculate BR_P and BR_F. If we modify a little bit of the weight values, the result is totally going to be changed. Actually, these weight values really depend on what users think. Remember the original equation for BR_P calculation is below:

$$BR\_P = max(\ \tfrac{A}{B})\cdot 100\cdot 30\% + \tfrac{C}{D}\cdot 100\cdot 30\% + \tfrac{E}{F}\cdot 100\cdot 40\% \tag{5.1}$$

If we change the weight value as follows:

$$BR\_P = max(\ \tfrac{A}{B})\cdot 100\cdot 45\% + \tfrac{C}{D}\cdot 100\cdot 30\% + \tfrac{E}{F}\cdot 100\cdot 25\% \tag{5.2}$$

Then, this second equation tells us that the user emphasizes more the worst comment the redditor posted. In the meantime, the degree of active of each user is going to be less emphasized. In other words, we mind less how the redditor usually comments, we are going to evaluate the worst comment he has ever done in order to give that redditor very high bad rate value.

Likewise, in case of calculating BR_F. If we do no mind the degree of active user, that is we do not classify redditors into different class. Then, the weight value is always equal to 1. Thus, BR_F= BR_P. Also, the BR_F value is going to change if we modify the weight value of each class of redditors.

Thus, with respect to the above point, it leads us to propose a possible future work, which is adding feedback relevance to the project.

Relevance feedback is a feature of some information retrieval systems. The idea behind relevance feedback is to take the results that are initially returned from a given query and to use information about whether or not those results are relevant to perform a new query.

We can apply feedback relevance concept as follows. After we show the result to the user which is a list of 20 worst behaviour redditors with the correspond "bad" comment. We could ask user to compare the comment of these 20 redittors and reorder the list. As we mentioned in the previous section, if the result changed that means the weight values is modified. Hence, we can use the new list to calculate what the current weight values are. Thus, we could intuitively know that whether the user emphasize more on the redditor's worst comment or the overall of its comments.

In fact, feedback relevance, this such charming feature was considered to be performed in this work. But due to various circumstance, I was not able to use it. However, it could be an appropriate future work to improve the project.

Appendix A

Prerequisite for program execution

Before executing the program, we need to do the following necessaries procedures.

We need firstly to describe the files contained in the src folder. Inside this folder, it has two Excel files, two IPython notebook files with the extension ".ipynb", one text file(".txt") and one Python file(".py"). Here, we focus on one of these IPython notebook files which contains the program code that we are going to execute. It is "TFG_Detecting_insults_words.ipynb".

The first step we need to do is to start the cluster which we have already seen the details in the section 2.2.4. Then, from the notebook interface we open the file "TFG_Detecting_insults_words.ipynb". The opened file will have various cells. We need to execute each cell with top-down approach by press "ctrl" and "enter" keys.

Before we execute the next cell, we need to wait the current cell finishes its execution.

After we execute all the cells in the notebook, the result will be shown as a DataFrame tabular inside the notebook. Also, it will be stored in one of the two Excel files we mentioned before which is "Final_result_top_20.xlsx".

Bibliography

[1] Big data. Documentation retrieved from Wikipedia https://en.wikipedia.org/wiki/Big_data [Online;].

[2] Ipython. Documentation retrieved from Wikipedia https://en.wikipedia.org/wiki/IPython#Parallel_computing [Online;].

[3] kaggle insults. The modified file which the original file is retrieved from GitHub https://github.com/amueller/kaggle_insults/ [Online;].

[4] Reddit. Documentation retrieved from Wikipedia https://en.wikipedia.org/wiki/Reddit [Online;].

[5] Using ipython for parallel computing. Documentation retrieved from IPython Documentaion https://ipython.org/ipython-doc/3/parallel/parallel_intro.html [Online;].

[6] L. Garrido F. Danti. "parallel computing" book chapter. Department of Mathematics and Computer Science. 2016.

[7] Sean McClure. Data science and big data:two very different beasts. http://www.kdnuggets.com/2015/07/data-science-big-data-different-beasts.html. [Online;].

[8] Gregory Piatetsky. Four main languages for analytics, data mining, data science. 2014. Documentation retrieved from http://www.kdnuggets.com/2014/08/four-main-languages-analytics-data-mining-data-science.html [Online;].

[9] Martijn Theuwissen. R vs python for data science: The winner is... Documentation retrieved from http://www.kdnuggets.com/2015/05/r-vs-python-data-science.html [Online;].